Claims

What is claimed is:

[c1] A method of transmitting a first message from a first process on a first node to a second process on a second node, the method comprising:

the first process defining a first process memory space on the second node, wherein the first process memory space is written to by the first process;

the first process assigning a separate postbox portion of the first process memory space for each respective process on the second node; and

the first process storing the first message in the first process memory space and a first message indicator in the postbox portion of the first memory space assigned to the second process.

[c2] The method of claim 1, wherein the first message indicator comprises location information of the first message in the first process memory space.

[c3] The method of claim 2, further comprising:

the second process polling its respective postbox portion of the first process memory space; and

upon the second process detecting the first message indicator in its respective postbox portion of the first process memory space, the second process retrieving the first message from the first process memory space dependent upon the location information in the first message indicator.

[c4] The method of claim 1, further comprising:

the first process determining whether the first message can be stored in the first process memory space.

[c5] The method of claim 4, further comprising:

the first process determining that the first message cannot be stored in the first process memory space if the first message is larger than a predetermined number of bytes.

[c6] The method of claim 1, further comprising:

the first process preparing a second message to send to a third process on the second node;

the first process determining whether the second message can be stored in the first process memory space; and

if the first process determines that the second message can be stored, the first process storing the second message in the first process memory space and a second message indicator in the postbox portion of the first process memory space assigned to the third process.

[c7] The method of claim 1, further comprising:

the first process maintaining a list of messages sent to other process; and

the first process adding information with respect to the first message to a list.

[c8] A method of transmitting a first message from a first process to a second process on a second node, a first process memory space being provided on the second node, the first process memory space comprising separate postbox portions for each respective process on the second node, wherein the first process memory space is written to by the first process, the method comprising:

the first process determining if the postbox portion for the second process can store a first message indicator for the first message;

if the postbox portion for the second process can store the first indicator for the first message, the first process determining if there is sufficient

unallocated first process memory space available to store the first message;

if there is sufficient unallocated first process memory space, the first process storing the first message in the first process memory space; and

storing the first message indicator in the postbox portion of the first process memory space assigned to the second process.

[c9] The method of claim 8, wherein the first message indicator comprises location information of the first message in the first process memory space.

[c10] The method of claim 8, further comprising:

if there is insufficient unallocated first process memory space available, the first process accessing a first list of second node processes to which the first process has sent messages to identify a portion of the first process memory space to be unallocated in order to make available first process memory space in which to store the first message.

[c11] The method of claim 8, further comprising:

if there is insufficient unallocated first process memory space available:

the first process identifying a portion of the first process memory space storing a second message that has already been retrieved by the second process; and

the first process making the identified portion of the first process memory space available to store the first message.

[c12] The method of claim 8, further comprising:

if there is insufficient unallocated first process memory space available:

the first process identifying a portion of the first process memory space storing a second message that has already been retrieved by a third

process on the second node to which the second message was sent; and

the first process making the identified portion of the first process memory space available to store the first message.

[c13] The method of claim 12, wherein the first process determines that the second message has been retrieved by the third process by determining that an acknowledgment message has been received from the third process with respect to the second message.

[c14] A computing node, comprising:

a first process memory space including a separate postbox portion for each process on the node, the first process memory space to be written to by a first process on a second node; and

means for each process on the computing node to access a respective postbox portion to retrieve a message indicator identifying a location in the first process memory space of a message from the first process to the respective process on the computing node.

[c15] The computing node of claim 14, further comprising a plurality of processors.

[c16] A method for managing memory when a process on a first node attempts to store a current message in a memory space on a second node, the method comprising:

determining whether any acknowledgment signals have been received by the first node from a receiver of a previously sent message that used memory in the memory space;

selectively reclaiming a postbox associated with the previously sent message if an acknowledgment signal has been received from the receiver of the previously sent message; and

storing the current message if the postbox associated with the previously

sent message is reclaimed.

[c17] The method of claim 16, further comprising:

determining whether the postbox contains a pointer to a buffer block; and

selectively deallocating the buffer block if the postbox contains a pointer to the buffer block.

[c18] The method of claim 17, further comprising:

decrementing a buffer count for the receiver of the previously sent message based on the freeing of the at least one buffer block.

[c19] The method of claim 18, wherein the buffer count is maintained to track a number of buffers blocks associated with the receiver of the previously sent message.

[c20] The method of claim 18, further comprising:

determining whether the buffer count for the receiver of the previously sent message is zero; and

selectively removing the receiver of the previously sent message from a receiver list if the buffer count for the receiver of the previously sent message is zero.

[c21] The method of claim 20, wherein the receiver list is maintained to track a plurality of receiving process.

[c22] A method for handling message transfers by selectively transferring a message in pieces, the method comprising:

determining whether a size of a remaining message for a receiving process is greater than a cyclic transaction size; and

storing the remaining message into a memory space if the size of the remaining message for the receiving process is not greater than the cyclic transaction size.

[c23] The method of claim 22, further comprising:

selectively breaking down the remaining message into smaller pieces if the size of the remaining message is greater than the cyclic transaction size.

[c24] The method of claim 23, wherein selectively breaking down the remaining message into smaller pieces is done such that memory used to send a piece of the remaining message can be reused to send a subsequent piece of the remaining message.

[c25] The method of claim 22, wherein the cyclic transaction size is the maximum amount of memory that can be used to send a message, and wherein the cyclic transaction size is the maximum amount of memory that can be used to send a piece of a message.

[c26] The method of claim 22, further comprising:

attempting to recover memory allocated to send a previous message to the first receiving process in order to send the remaining message to the first receiving process; and

using the memory allocated to send the previous message if the attempt is successful.

[c27] The method of claim 22, further comprising:

maintaining a receiver list to track a plurality of receiving processes, and wherein the plurality of receiving processes on the receiver list are maintained in an order in which the plurality of receiving process es were communicated to by a sending process.

[c28] The method of claim 27, further comprising:

attempting to recover memory used by the plurality of receiving process es by

traversing the receiver list; and

selectively recovering memory from those receiving processes on the receiver list that reside on a node which is the same as the node on which the first receiving process resides.

[c29] The method of claim 27, further comprising:

counting a number of buffer pool blocks used by each of the plurality of receiving processes on the receiver list.

[c30] The method of claim 27, wherein individual receiving processes of the plurality of receiving processes are removed from the receiver list if those individual receiving processes do not consume any buffer pool blocks.

[c31] A method for creating and handling a data memory segment and an acknowledgment segment when a first process on a first node needs to send a message to a second process on a second node, the method further comprising:

determining whether there is a data memory segment for the first process on the second node;

creating the data memory segment for the first process on the second node if there is not already a data memory segment for the first process on the second node;

determining whether there is an acknowledgment segment on the first node; and

creating the acknowledgment segment on the first node if there is not already an acknowledgment segment on the first node.

[c32] The method of claim 31, wherein the data memory segment is used to store the message, and wherein the acknowledgment segment is used by the second process to indicate to the first process that the second process has received the message.

[c33] The method of claim 31, wherein the acknowledgment segment is written into by the second process and read from by the first process.

[c34] The method of claim 31, wherein creating the data memory segment is done when the first process needs to send a message to the second process.

[c35] The method of claim 31, wherein creating the acknowledgment segment is done when the second process needs to indicate to the first process that the second process has received the message.

[c36] The method of claim 31, wherein the acknowledgment segment is selectively divided into m sections, and wherein m represents a number of processes on the first node.

[c37] The method of claim 36, wherein each of the m sections is selectively divided into n acknowledgment blocks, and wherein n represents a number of processes on the second node.

[c38] The method of claim 31, wherein the acknowledgment segment is used by a plurality of processes on the second node to send acknowledgment indications to the first node.

[c39] The method of claim 31, wherein the acknowledgment segment is created on each node in a pair of nodes.

[c40] The method of claim 31, further comprising:

selectively establishing a simplex connection between the first process and the second process, wherein the simplex connection is a one way communication channel between the first process and the second process.

[c41] The method of claim 31, further comprising:

sending a second message from the first process on the first node to a third process on a third node.